**MINI PROJECT:**

**Nayyar Abbasi & Abdul Raheem**

**21F-9173 & 21F-9051**

**CODE:**

#!/bin/bash

# Define the board size and dimensions

board\_size=5

total\_tiles=$((board\_size \* board\_size))

blank\_tile=" "

# Initialize the game board and goal state

declare -a game\_board

declare -a goal\_state

# Function to initialize the game board with a random solvable state

initialize\_board() {

while true; do

local tiles=($(seq 1 $total\_tiles))

tiles=($(shuf -e "${tiles[@]}"))

game\_board=("${tiles[@]}")

local blank\_index=$((RANDOM % total\_tiles))

game\_board[$blank\_index]="$blank\_tile"

if Is\_Solvable; then

break

fi

done

}

# Function to shuffle the game board (no longer used)

shuffle\_board() {

echo "Shuffle not needed"

}

# Function to print the game board

print\_board() {

for ((i = 0; i < total\_tiles; i++)); do

printf "[%2s] " "${game\_board[$i]}"

if ((i % board\_size == board\_size - 1)); then

echo

fi

done

}

# Function to make a move (e.g., "U" for up, "D" for down, "L" for left, "R" for right)

make\_move() {

local move=$1

local blank\_index

for ((i = 0; i < total\_tiles; i++)); do

if [[ "${game\_board[$i]}" == "$blank\_tile" ]]; then

blank\_index=$i

break

fi

done

local row=$((blank\_index / board\_size))

local col=$((blank\_index % board\_size))

case $move in

"U")

if ((row > 0)); then

local new\_index=$((blank\_index - board\_size))

game\_board[$blank\_index]=${game\_board[$new\_index]}

game\_board[$new\_index]="$blank\_tile"

fi

;;

"D")

if ((row < board\_size - 1)); then

local new\_index=$((blank\_index + board\_size))

game\_board[$blank\_index]=${game\_board[$new\_index]}

game\_board[$new\_index]="$blank\_tile"

fi

;;

"L")

if ((col > 0)); then

local new\_index=$((blank\_index - 1))

game\_board[$blank\_index]=${game\_board[$new\_index]}

game\_board[$new\_index]="$blank\_tile"

fi

;;

"R")

if ((col < board\_size - 1)); then

local new\_index=$((blank\_index + 1))

game\_board[$blank\_index]=${game\_board[$new\_index]}

game\_board[$new\_index]="$blank\_tile"

fi

;;

esac

}

# Function to randomly initialize a goal state, ensuring it is reachable from the initial state

goal() {

local tiles=($(seq 1 $((total\_tiles - 1))) " ")

goal\_state=("${tiles[@]}")

# Check if the goal state is solvable; if not, swap two tiles to make it solvable

if !Is\_Solvable; then

# Swap the last two tiles

local temp=${goal\_state[$((total\_tiles - 1))]}

goal\_state[$((total\_tiles - 1))]=${goal\_state[$((total\_tiles - 2))]}

goal\_state[$((total\_tiles - 2))]=$temp

fi

}

# Function to determine if the goal state is reachable from the initial state

Is\_Solvable() {

local inv\_initial=$(count\_inversions "${game\_board[@]}")

local inv\_goal=$(count\_inversions "${goal\_state[@]}")

if ((board\_size % 2 == 1)); then

# Odd board size

return $(((inv\_initial + inv\_goal) % 2))

else

# Even board size

local initial\_blank\_row=$((board\_size - (inv\_initial / board\_size) - 1))

local goal\_blank\_row=$((board\_size - (inv\_goal / board\_size) - 1))

local row\_diff=$((goal\_blank\_row - initial\_blank\_row))

return $(((inv\_initial + inv\_goal + row\_diff) % 2))

fi

}

# Helper function to count inversions

count\_inversions() {

local -a arr=("$@")

local inversions=0

for ((i = 0; i < total\_tiles - 1; i++)); do

for ((j = i + 1; j < total\_tiles; j++)); do

if [[ ${arr[i]} -gt ${arr[j]} && ${arr[i]} != "$blank\_tile" && ${arr[j]} != "$blank\_tile" ]]; then

((inversions++))

fi

done

done

echo $inversions

}

# Function to check if the current state matches the goal state

is\_goal() {

local -a state=("${game\_board[@]}")

if [ "${#state[@]}" -ne "${#goal\_state[@]}" ]; then

return 1 # Array lengths do not match

fi

for ((i = 0; i < total\_tiles; i++)); do

if [[ ${state[i]} != ${goal\_state[i]} ]]; then

return 1 # Not in the goal state

fi

done

return 0 # In the goal state

}

# Function to get legal moves for the current state

legal\_moves() {

local -a state=("${game\_board[@]}")

local moves=()

for ((i = 0; i < total\_tiles; i++)); do

if [[ "${state[i]}" == "$blank\_tile" ]]; then

local row=$((i / board\_size))

local col=$((i % board\_size))

if ((row > 0)); then

moves+=("U")

fi

if ((row < board\_size - 1)); then

moves+=("D")

fi

if ((col > 0)); then

moves+=("L")

fi

if ((col < board\_size - 1)); then

moves+=("R")

fi

fi

done

echo "${moves[@]}"

}

# Function to print the sequence of moves taken to reach the goal state

print\_path() {

local -a state=("${game\_board[@]}")

local path=()

while !is\_goal "${state[@]}"; do

for move in $(legal\_moves "${state[@]}"); do

local new\_state=("${state[@]}")

make\_move "$move"

if ! is\_goal "${new\_state[@]}"; then

state=("${new\_state[@]}")

path+=("$move")

break

fi

done

done

echo "Initial State: ${game\_board[@]}"

echo "Goal State: ${goal\_state[@]}"

echo "Path Taken: ${path[@]}"

}

# Main game loop

initialize\_board

goal

while true; do

clear

echo " GOAL STATE"

echo " 1 2 3 4 5"

echo " 6 7 8 9 10"

echo " 11 12 13 14 15"

echo " 16 17 18 19 20"

echo " 21 22 23 24 "

echo " "

echo "24-Puzzle Game:"

echo "Enter a move (U/D/L/R) or Q to quit:"

print\_board

read -n 1 move

if [[ "$move" == "Q" || "$move" == "q" ]]; then

break

fi

make\_move "$move"

if is\_goal; then

clear

echo "Congratulations! You solved the puzzle."

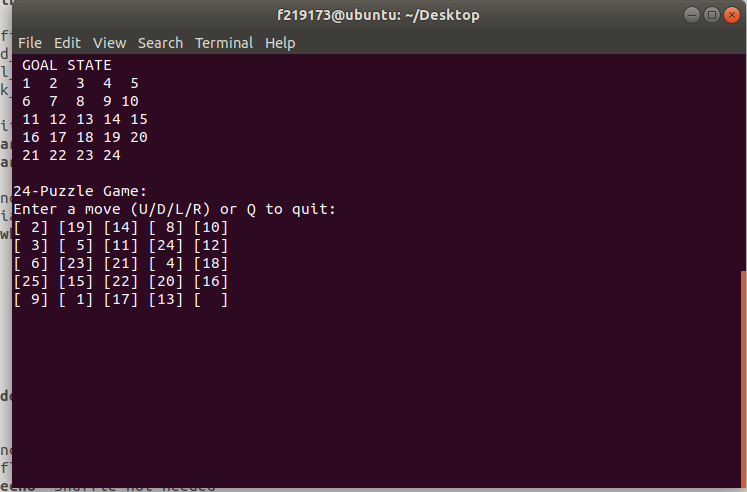
print\_board

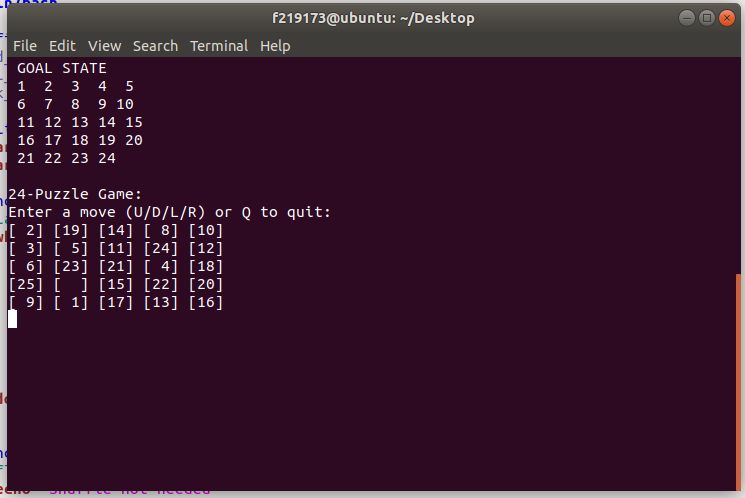
break

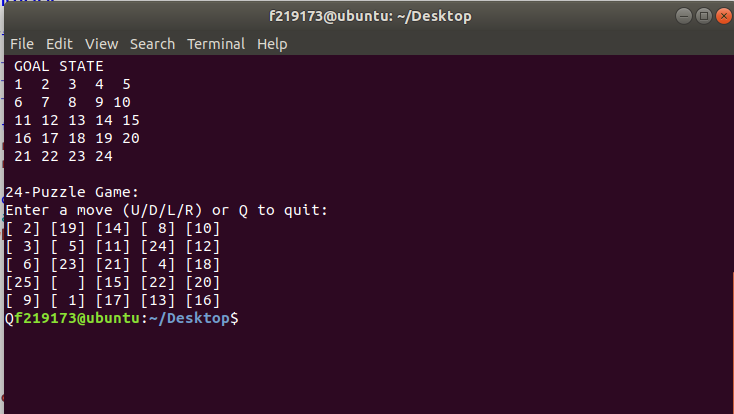
fi

done

**OUTPUT:**

****

****

****